Microsoft TechNet

Microsoft Windows CE Graphics Features

Introduction

In the Microsoft® Windows® CE operating system, as in the other Microsoft Win32® operating systems, the subsystem that controls the display of text and graphics is called the graphics device interface (GDI). Like its desktop counterparts, the Windows CE GDI draws lines, curves, closed figures, text, and bitmapped images on display devices and printers.

In version 2.0 of Windows CE, GDI functionality has been expanded to include many of the advanced graphical features available in other Win32 application programming interfaces (API). No longer is world-class graphical display limited to desktop computers; GDI allows Windows CE-based devices to create graphics that are both attractive and useful.

This paper discusses each major feature of GDI. Special attention is given to features that have been added or substantially modified since Windows CE version 1.0. The paper also discusses how each feature differs, if at all, from the standard Win32-based application programming interface.

In this paper, GDI will refer only to the Window CE subsystem, not to the equivalent subsystem of other Windows-based platforms.

↑ Top of page

GDI Features

Device Contexts

A device context (DC) is a structure that GDI uses to store information about the graphical objects that govern the appearance of text and graphics on a display or printing device. The type of pen, brush, font, colors, and bitmap that will be used to create a display on a particular device are all stored in a device context.

Windows CE supports the device contexts described in the following table.

Device Context Type	Purpose
Display	Supports drawing operations on a video display.
Printer	Supports drawing operations on a printer.
Memory	Supports drawing operations on a device-independent bitmap.

An application can never access a device context directly; it operates on the structure indirectly by calling functions that have access to the DC.

A handle to a device context can be obtained by calling the **BeginPaint** function. Device contexts can be retrieved by the **GetDC** function and created with the **CreateDC** and **CreateCompatibleDC** functions.

Version 2.0 Additions

Memory device contexts have been improved in version 2.0 so that they are now fully compatible with display DCs. Print DCs are new to version 2.0.

Differences from Other Win32-based Platforms

Windows CE does not support information device contexts.

Shape Drawing & Filling

In addition to simple line drawing, GDI can draw a wide set of shapes using the following functions:

Function Name	Purpose
Ellipse	Draws and fills a ellipse.
Polygon	Draws and fills a user-defined shape.
Polyline	Draws a line formed from a series of connected straight line segments. The resulting polyline can approximate almost any shape.
Rectangle	Draws and fills a rectangle.
RoundRect	Draws and fills a rectangle with rounded corners.

Version 2.0 Additions

In version 2.0, the polygon function has been modified so that it correctly fills concave shapes and self-intersecting polygons. A self-intersecting polygon is a polygon whose exterior lines cross one another when the polygon is drawn. Differences from other Win32-based platforms GDI does not support the creation of paths or non-rectangular regions, nor does it support the **LineTo** and **MoveTo** functions. In addition, GDI does not support the functions necessary to create the following shapes and lines.

- Arcs
- Bezier curves
- Chords
- Pies
- PolyPolygons
- PolyPolylines

Pens & Brushes

Pens are the graphics objects GDI uses to draw lines; brushes are the graphical objects it uses to fill the interiors of closed shapes.

Pens

The BLACK_PEN and WHITE_PEN each draws a solid, one-pixel-wide line in its respective color. These pens, along with NULL_PEN, which does not draw, are the stock GDI pens. The GetStockObject function can be used to select a stock pen.

The CreatePen and CreatePenIndirect functions can be used to create pens that have different attributes than the stock pens. These functions allow the user to define a pen's line width, color, and pen style. Possible pen styles include the following:

Pen Style	Description
PS_SOLID	Draws a solid line.
PS_DASH	Draws a dashed line.

PS_NULL	Does not draw a line.

Brushes

Stock brushes, like stock pens, can be selected with the **SelectObject** function. The **CreateDIBPatternBrushPt** function can be used to create a brush of any size. Pattern brushes can be any color or combination of colors.

Version 2.0 Additions

Color pens are new to version 2.0, as are color and multisize pattern brushes.

Differences from Other Win32-based Platforms

GDI supports wide pens and dashed pens, but it does not support wide dashed pens. Dotted pens and inside frame pens are not supported, nor are any user-specified endcap styles (e.g., PS_ENDCAP_ROUND).

Hatched brushes are not supported in Windows CE. However, the effect of a hatched brush can be achieved by creating a custom brush with the CreateDIBPatternBrushPt function.

Windows CE supports multisize pattern brushes, which are also supported by the Microsoft Windows NT® operating system. The Microsoft Windows 95 operating system only supports the standard 8-pixel by 8-pixel brushes.

Bit Block Transfer Functions

Bit block transfer functions (blts) are used to transfer and/or alter bitmaps. There are four kinds of blt functions, **PatBlt**, **BitBlt**, **MaskBlt**, and **StretchBlt**, plus a related function called **TransparentImage**.

The blt functions make heavy use of bitwise Boolean pixel operations, more commonly termed raster operation codes (ROPS). The following charts describe the ROP codes and the blt functions that use them.

ROP Type	**Description**
ROP2	Combines a pen or brush with a destination bitmap in one of 16 possible combinations.
ROP3	Combines a brush, a source bitmap, and a destination bitmap in one of 256 possible combinations.
ROP4	Uses a binary "mask" bitmap to combine a foreground ROP3 and a background ROP3. The mask uses 0s and 1s to indicate the areas where each ROP3 will be used.
BLT Function	**Description**
PatBlt	Paints a selected rectangle using a selected brush and an ROP3 code.
BitBlt	Paints a selected rectangle using a selected brush, a source bitmap, and an ROP3 code.
MaskBlt	Paints a given rectangle using two bitmaps and an ROP4 code.
StretchBlt	Copies a bitmap from a source rectangle into a destination rectangle, stretching or compressing the bitmap to fit the destination rectangle.
TransparentImage	Makes a copy of a bitmap, omitting the portions drawn in a transparent color.

Taken together, the blt functions and ROP codes provide the applications programmer with a powerful and extremely useful set of tools for moving, combining, and transforming bitmaps.

Version 2.0 Additions

Version 1.0 supported only a handful of ROP codes; version 2.0 supports them all. **StretchBlt** has been improved so that stretched images more closely resemble the images on which they are based. Image inversion, which was not possible in version 1.0, is fully supported in version 2.0. **TransparentImage** is new to version 2.0.

Colors

One of the most exciting developments in version 2.0 of Windows CE is the addition of color. The new Windows CE GDI supports the full range of colors available in other 32-bit Windows-based platforms.

The color range available to a display device or operating system is determined primarily by the pixel depth that it supports. The pixel depth is measured in bits per pixel (bpp). Each bit can have a value of 1 or 0. A display that supports only one bit per pixel allows only two values, black and white. A pixel depth of 2 bpp allows four possible values (all possible combinations of 0s and 1s with two bits): black, white, light gray, and dark gray. In general, the number of possible colors is equal to 2 raised to the power of the pixel depth.

The new Windows CE GDI supports pixel depths of 1, 2, 4, 8, 16, 24, and 32 bpp. The higher-end pixel depths allow full-color displays, while the lower-end pixel depths support monochrome display devices and applications.

The GDI color system is not only powerful, it is extremely versatile. It allows bit block transfer functions between bitmaps with different pixel depths. This is possible because of GDI's use of device-independent bitmaps (DIB). DIBs are bitmaps that are stored internally, independent of any particular display or printing device. DIBS contain their own color tables and can be displayed and/or printed in a variety of color formats, depending on the capabilities of the display or printing hardware. Virtually all GDI graphics information is stored in DIB format.

Version 2.0 Additions

Colors are new to version 2.0. Version 1.0 only supported pixel depths of 1 and 2 bpp.

Differences from Other Win32-based Platforms

Windows CE supports pixel depths of 2 bpp, which are not supported in other 32-bit Windows platforms.

Windows CE does not support dithering.

Windows CE does not support compressed bitmap formats, such as run-length encoded bitmaps.

Palettes

GDI allows a Windows CE-based application and a display device to jointly determine the optimum strategy for graphical display. Applications should use the **GetDeviceCaps** function to assess the display characteristics of the display device. The application can then create a device context based on the optimum display strategy for that device.

GDI supports both palettized and non-palettized color display devices. Palettized devices contain their own built-in color palette in their display card; GDI reads the value encoded in the device's display card and looks up the corresponding color in the device's color table.

Non-palettized devices have the color encoded directly in the frame buffer pixel. They use the pixels' bit value to directly define the color without the need of a hardcoded palette.

Once the color capabilities of the device context have been determined, the application can create a palette using the **CreatePalette** function, and then select it into the current device context with the **SelectPalette** function. The palette created and selected by these two functions is a logical palette, which exists apart from the system palette used by the display device. The **RealizePalette** function makes the system palette assume the values in the currently selected logical palette.

Version 2.0 Additions

Color palettes, and their related functions, are new to version 2.0.

Differences from Other Win32-based Platforms

From a programmer's perspective, one of the most crucial differences between Windows CE and other Win32 operating systems is that Windows CE does not arbitrate between the palettes of the background and foreground applications. The application running in the foreground has complete control over the system palette. Because of this, it is highly recommended that Windows CE-based applications confine themselves to the first ten and last ten colors included in the stock palette - the so-called "Windows Colors." Applications that use these colors should display properly while in the background; applications that use other colors may not.

Fonts

GDI supports both TrueType fonts® and raster fonts, but allows only one to be used on any given platform.

The outlines of TrueType fonts are defined not by pixel patterns, as are raster fonts, but by an encoded set of lines and curves. TrueType fonts can be readily rescaled and even rotated. GDI supports all standard TrueType font files for Windows.

Version 2.0 Additions

TrueType fonts are new to version 2.0. Version 1.0 supported raster fonts only.

Printing

GDI supports full graphical printing.

Version 2.0 Additions

Printing is new to version 2.0 of Windows CE.

Differences from Other Win32-based Platforms

Windows CE does not have a print manager, nor does it support spooling or allow multiple copies to be printed. Printing is single-threaded in that only one application can print at a time.

Display Characteristics

Windows CE supports clipping regions, which allows applications to restrict their output to a subregion of the client area. Clipping areas must be selected into the device context associated with a given device.

Differences from Other Win32-based Platforms

Multiple mapping modes are not supported in Windows CE. GDI supports only the text mapping mode (called the MM-TEXT mode on desktop Win32-based platforms), which maps a bitmap to a display device the same way text is read - from left to right and top to bottom. The viewpoint origin, which is the upper-left corner of the display area, can be changed with the **SetViewPointOrg** function.

Tips & Tricks

Applications that will be primarily run on 2-bpp or 4-bpp display devices should avoid using 8-bpp bitmaps whenever possible, since the color conversions will slow down blt operations.

The iUsage parameter should always be set to **DIB_RGB_COLORS** in all the functions in which it appears. It can be set to **DIB_PAL_COLORS**, when an 8-bpp bitmap is being used, however, the values in the bmiColors array member of the **BITMAPINFO** structure will be ignored.

↑ Top of page

Summary

Windows GDI is a full-fledged graphics system that supports the following features.

GDI FEATURE	Supported Attributes
Shape Drawing	Ellipse, Polygon, Polyline, Rectangle, RoundRect
Pens & Brushes	Dashed, Dotted, Wide, and Solid Pens; Multisize Brushes
Bit Block Transfer Functions	PatBlt, BitBlt, MaskBlt, StretchBlt, TransparentImage
ROP Codes	All ROP2, ROP3, and ROP4 Codes
Colors	Pixel depths of 1, 2, 4, 8, 16, 24, and 32 bbp.
Palettes	Palettized and Non-Palettized Device Support
Fonts	TrueType and Raster Fonts
Printing	Graphical Printing

The functionality of Windows CE, including GDI, is not as extensive as its Win32-based desktop counterparts - nor is it intended to be. Display and printing functions not needed by Windows CE-type devices have been eliminated from GDI. As a consequence, GDI is a powerful, full-color graphical display system with a relatively small footprint. Its small size and versatility make it capable of displaying text and graphics on a wide spectrum of Windows CE-type devices - some of which have not even been imagined yet.

Top of page